AMENDMENTS TO THE CLAIMS

1. (Previously Presented) A method for processing data stored in a memory shared among a plurality of processors, comprising:

providing a semaphore associated with a first portion of the memory;

storing tasks in the first portion of the memory, the tasks respectively related to information associated with a second portion of the memory;

determining a state of the semaphore;

controlling access among the plurality of processors to the first portion of the memory in response to the state of the semaphore; and

executing a task to process at least some of the information within the second portion of the memory in response to a processor of the plurality of processors gaining access to the first portion of the memory;

providing another memory respectively coupled to each of the plurality of processors; and

storing a program in the other memory accessible by each of the plurality of processors, the program capable of executing at least one of the tasks.

2. (Original) The method of claim 1, wherein the controlling comprises:

allowing one of the plurality of processors to access the first portion of the memory responsive to the semaphore having a first state value; and

blocking access to all of the plurality of processors to the first portion of the memory responsive to the semaphore having a second state value.

Claim 3. (Cancelled)

4. (Original) The method of claim 1, further comprising coupling the semaphore to a bus shared by each of the plurality of processors.

5. (Original) The method of claim 1, wherein the storing comprises:

initializing state of the semaphore to allow access to the first portion of the

memory; and

receiving tasks to be executed by the processor of the plurality of processors.

6. (Previously Presented) A method for processing data stored in a memory shared among a plurality of processors, comprising:

storing task data for one or more tasks;

relating each of the one or more tasks to respective one or more data segments stored in a second portion of the memory;

associating a semaphore with the task data;

controlling access among the plurality of processors to the task data in response to a state of the semaphore; and

executing a task of the one or more tasks to process a respective data segment in response to a processor of the plurality of processors gaining access to the task data;

providing another memory respectively coupled to each of the plurality of processors; and

storing a program in the other memory of each of the plurality of processors, the program capable of executing the task of the one or more tasks.

Claim 7. (Cancelled)

8. (Original) The method of claim 6, wherein the controlling comprises:

allowing one of the plurality of processors to access the task data responsive to the semaphore having a first state value; and

blocking access to all of the plurality of processors to the task data responsive to the semaphore having a second state value.

9. (Original) The method of claim 6, wherein the task data comprises a current task of the one or more tasks, a next task to be executed immediately following the current task, and a number of idle processors of the plurality of processors.

10. (Original) The method of claim 6, further comprising storing an identifier for the processor that executed the task.

11. (Original) The method of claim 6, wherein each of the one or more tasks is related to an identifier indicative of task-type.

12. (Original) The method of claim 6, wherein the relating comprises:

storing a respective address of the data segment within the second portion of the memory for each of the one or more tasks; and

storing a length of the data segment respectively associated with each of the one or more tasks.

Claims 13-20. (Cancelled)